Amendments to the Claims:

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

Claim 1 (previously presented): A method for processing software instructions comprising:

decomposing a macroinstruction into a plurality of microinstructions;

determining whether at least two of the plurality of microinstructions are required to issue in parallel;

forcing the parallel issue of at least two of the plurality of microinstructions simultaneously regardless of conflict checking;

executing all of the plurality of microinstructions simultaneously, in lockstep using functional units in a floating point unit;

determining whether an exception occurs in any of the microinstructions, before writing results of the executing to result registers wherein the determining step is performed prior to any writing step and the method does not write any results to temporary registers;

if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the results of the executing from being written to the result registers; and

if no exception occurs in any of the microinstructions, writing the results of the executing to the result registers.

Claim 2 (canceled).

Claim 3 (previously presented): The method of claim 1, wherein the microinstructions are executed on separate execution units, but appear as though they were executed on a single execution unit.

Claim 4 (previously presented): The method of claim 1, wherein all of the microinstructions are executed on the same clock cycle.

Claim 5 (previously presented): The method of claim 1, wherein the microinstructions are executed over multiple clock cycles.

Claim 6 (canceled).

Claim 7 (previously presented): The method of claim 1, wherein the system allows a single instruction to operate on multiple single-precision floating-point values.

Claim 8 (previously presented): The method of claim 1, further comprising updating a flag based upon a result of the execution of the microinstructions.

Claim 9 (previously presented): The method of claim 1, further comprising,

if an unmasked exception occurs, canceling the execution of all of the plurality of microinstructions, without regard to the relative ages of each of the plurality of microinstructions, and invoking a microcode handler, and

if an unmasked exception does not occur, updating at least one exception flag by independently generating a logical OR of exceptions for a plurality of functional units.

Claim 10 (previously presented): A method for processing software instructions comprising:

providing two microinstructions to emulate a high-half and a low-half Streaming Single Instruction Multiple-Data Extensions (SSE) operation;

determining whether the two microinstructions are required to issue in parallel;

forcing the high-half and low-half operations to issue in parallel regardless of conflict checking;

dispatching the high-half and low-half operations simultaneously to a first floating point unit and to a second floating point unit, respectively;

executing the high-half and low-half operations simultaneously, in lockstep;;

generating a signal from an emulator's hardware;

sending the signal to the first and second floating point functional units;;

determining whether an exception is taken in either the first or the second floating point unit, wherein the determining step is performed prior to any writing step and the method does not write any results to temporary registers;

if an exception is taken in either the first or second floating point unit,

preventing results from the high-half and low-half operations from being written to result registers and

canceling both the high-half and low-half operations; and

updating MXCSR flags based upon the results of the first and second floating point units.

Claim 11 (previously presented): The method of claim 10, wherein the preventing and canceling steps do not depend upon the relative ages of the two microinstructions.

Claim 12 (previously presented): A computer system comprising:

a processor comprising:

a floating point unit comprising a plurality of functional units adapted to execute microinstructions;

a ROM;

a plurality of floating point registers;

wherein the processor is configured to emulate an instruction set by:

decomposing a macroinstruction into a plurality of microinstructions;

determining whether at least two of the plurality of microinstructions are required to issue in parallel;

forcing the parallel issue of at least two of the plurality of microinstructions simultaneously to the functional units regardless of conflict checking;

determining whether an exception occurs in any of the functional units, wherein the determining step is performed prior to any setting step and the method does not set any temporary registers;

setting result registers for results of each of the functional units only if no exception occurs in any of the functional units; and

if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the setting of result registers for all of the functional units.

Claim 13 (previously presented): The computer system of claim 12, wherein the processor is further configured to emulate the instruction set by executing all of the microinstructions.

Claim 14 (previously presented): The computer system of claim 13, wherein the microinstructions are executed on separate execution units, but appear as though they were executed on a single execution unit.

Claim 15 (previously presented): The computer system of claim 14, wherein the processor is further configured to emulate an instruction set by updating a flag based upon a result of the execution of the microinstructions.

Claims 16-17 (canceled).

Claim 18 (previously presented): The computer system of claim 12, further comprising an floating point register having 82 bits, wherein the computer system uses two floating point registers to emulate four 32-bit single-precision, floating point values in a Streaming Single Instruction Multiple-Data Extensions (SSE) register.

Claim 19 (previously presented): The method of claim 1, wherein the step of issuing comprises forcing the microinstructions to issue simultaneously, in lockstep with each other, and wherein the step of canceling comprises canceling all of the plurality of microinstructions without regard to the relative ages of the microinstructions and without using a backoff mechanism.

Claim 20 (canceled).

Claim 21 (previously presented): The method of claim 1, wherein the step of executing comprises executing using a plurality of functional units of a floating point unit, further comprising:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating a Streaming Single Instruction Multiple-Data Extensions (SSE) instruction; and

sending the signal to the functional units, and

wherein the step of determining comprises determining after the signal is sent.

Claim 22 (previously presented): The system of claim 12, wherein the processor is further configured to emulate an instruction set by:

generating a signal using emulation hardware, wherein the signal indicates that the functional units are emulating a Streaming Single Instruction Multiple-Data Extensions (SSE) instruction; and

sending the signal to the functional units, and

wherein the step of determining comprises determining after the signal is sent.

Claims 23-24 (canceled).

Claim 25 (previously presented): The method of claim 1, further comprising:

delaying issue of a first microinstruction if it is determined that said first microinstruction is required to be issued in parallel with at least one subsequent instruction that is not yet ready for execution.

Claim 26 (previously presented): A method for processing software instructions comprising:

in an in-order execution machine, decomposing a macroinstruction into a plurality of microinstructions;

forcing the parallel issue of at least two of the plurality of microinstructions simultaneously regardless of conflict checking;

executing all of the plurality of microinstructions simultaneously, in lockstep using functional units in a floating point unit;

determining whether an exception occurs in any of the microinstructions, before writing results of the executing to result registers wherein the determining step is performed prior to any writing step and the method does not write any results to temporary registers;

if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the results of the executing from being written to the result registers; and

if no exception occurs in any of the microinstructions, writing the results of the executing to the result registers.

Claim 27 (previously presented): The method of claim 26, further comprising:

determining whether at least two of the plurality of microinstructions must be issued in parallel.

Claim 28 (previously presented): A computer system comprising:

a processor comprising:

a floating point unit comprising a plurality of functional units adapted to execute microinstructions;

a ROM;

a plurality of floating point registers;

wherein the processor is configured to emulate an instruction set by:

in an in-order execution machine, decomposing a macroinstruction into a plurality of microinstructions;

forcing the parallel issue of at least two of the plurality of microinstructions simultaneously to the functional units regardless of conflict checking;

determining whether an exception occurs in any of the functional units, wherein the determining step is performed prior to any setting step and the method does not set any temporary registers;

setting result registers for results of each of the functional units only if no exception occurs in any of the functional units; and

if an exception occurs in any of the microinstructions, canceling all of the microinstructions and preventing the setting of result registers for all of the functional units.

Claim 29 (previously presented). The computer system of claim 28, wherein the processor is further configured to emulate an instruction set by:

determining whether at least two of the plurality of microinstructions must be issued in parallel.